Alexander BARKALOV, Larysa TITARENKO, Jacek BIEGANOWSKI
INSTYTUT INFORMATYKI I ELEKTRONIKI, UNIWERSYTET ZIELONOGÓRSKI

Metoda syntezy mikroprogramowanego układu sterującego z rozszerzonym formatem mikroinstrukcji

prof. dr hab. Alexander BARKALOV

Prof. Alexander A. Barkalov w latach 1976-1996 był pracownikiem dydaktycznym w Instytucie Informatyki Narodowej Politechniki Donieckiej. Współpracował aktywnie z Instytutem Cybernetyki im. V.M. Glushkova w Kijowie, gdzie uzyskał tytuł doktora habilitowanego ze specjalnością informatyka. W latach 1996-2003 pracował jako profesor w Instytucie Informatyki Narodowej Politechniki Donieckiej. Od 2004 pracuje jako profesor na Wydziale Elektrotechniki, Informatyki i Telekomunikacji Uniwersytetu Zielonogórskiego.

e-mail: A.Barkalov@iie.uz.zgora.pl

mgr inż. Jacek Bieganowski

Ukończył w roku 2003 studia na kierunku informatyka o specjalności inżynieria komputerowa. Od 2004 roku pracuje w Instytucie Informatyki i Elektroniki Uniwersytetu Zielonogórskiego na stanowisku asystenta. Jego zainteresowania związane są z programowaniem, systemami operacyjnymi, techniką cyfrową oraz algorytmami ewolucyjnymi.

e-mail: J.Bieganowski@iie.uz.zgora.pl

dr hab. Larysa TITARENKO, prof. UZ

Dr hab. Larysa Titarenko w 2004 roku obroniła rozprawę habilitacyjną i uzyskała tytuł doktora habilitowanego ze specjalnością telekomunikacja. W latach 2004-2005 pracowała jako profesor w Narodowym Uniwersytecie Radioelektroniki w Charkowie. Od 2005 pracuje na Wydziale Elektrotechniki, Informatyki Telekomunikacji Uniwersytetu Zielonogórskiego.

e-mail: L.Titarenko@iie.uz.zgora.pl

Streszczenie

W artykule przedstawiono metodę syntezy mikroprogramowanego układu sterującego ze wspólną pamięcią i rozszerzonym formatem mikroinstrukcji. Metoda jest zorientowana na zmniejszenie rozmiaru układu adresowego poprzez umieszczenie kodów klas łańcuchów pseudorównoważnych w pamięci sterującej. Uzyskuje się w ten sposób uproszczenie funkcji przejść części adresowej układu, co przekłada się na redukcję zasobów sprzętowych potrzebnych do implementacji jednostki sterującej w układach programowalnych typu CPLD i FPGA. W artykule zamieszczono wprowadzenie teoretyczne, przykład zastosowanie metody oraz wyniki badań uzyskane podczas syntezy testowych sieci działań przy użyciu oprogramowania Xilinx ISE 10.2 dla układów Xilinx Virtex II. Na postawie uzyskanych wyników można stwierdzić, że dla liniowych sieci działań uzyskuje się średnią redukcję rozmiaru układu na poziomie około 50% w porównaniu do podstawowego wariantu mikroprogramowanego układu sterującego.

Słowa kluczowe: mikroprogramowany układ sterujący, CPLD, FPGA

Synthesis method of CMCU with extended microinstruction format

Abstract

The paper presents new research results of synthesis method of Compositional Microprogram Control Unit (CMCU) with Common Memory and Extended Microinstructions for programmable logic devices such as CPLD and FPGA. Programmable logic devices are nowadays widely used for implementation of Control Units (CU) [3]. The problem of the optimization of CU is still actual in computer science and it solution permits to decrease the cost of the system [2]. The proposed method is oriented on reduction of hardware amount of CMCU addressing circuit by placing pseudoequivalent class codes in the control memory. These classes are formed by division of the set of Operational Linear Chains (OLC) into partitions which correspond to pseudoequivalent states of Moore FSM [2]. When class codes are stored in control memory the transition function is simplified and the hardware amount of addressing circuit is reduced when compared to CMCU base structure. The method can be applied when control algorithm to be implemented is linear i.e. the number of operational vertices exceeds the 75% of total number of vertices of Graph Scheme of Algorithm (GSA) to be implemented. The research results shows that application of the method for tested GSAs gives on average 50% decrease of hardware amount when compared to CMCU base structure (Tab. 4). The results were obtained in Xilinx ISE. The CMCU models were generated by our software and described in VHDL.

Keywords: compositional microprogram control unit, CMCU, CPLD, FPGA

1. Wstęp

Pomimo rosnącej pojemności logicznej układów programowalnych (CPLD, FPGA) nadal aktualnym problemem jest optymalizacja rozmiaru systemu cyfrowego. W szczególności optymalizacja układu sterującego będącego jednym z najważniejszych elementów systemu cyfrowego. Zagadnienie to jest istotne, ponieważ zmniejszenie rozmiaru układu sterującego przekłada się na redukcję kosztów projektowanego systemu [1].

Jednym ze sposobów pozwalających na redukcję zasobów sprzętowych jednostki sterującej jest dostosowanie jej struktury do charakterystyki implementowanego algorytmu sterującego [2]. Na przykład algorytm sterujący o charakterze liniowym (zawierający dużo bezwarunkowych przejść) może być zrealizowany z wykorzystaniem mikroprogramowanego układu sterującego (CMCU) [2]. Rozmiar układu (liczba komórek PAL, LUT) może zostać zredukowany poprzez zmniejszenie liczby składników funkcji Boolowskiej w dysjunkcyjnej postaci normalnej [3]. W przypadku układów CMCU można w ten sposób uprościć funkcję przejść realizowaną przez układ adresowy. Uproszczenie polega na zastąpieniu adresu mikroinstrukcji, podanego na wejście układu adresowego, kodem klasy łańcuchów pseudorównoważnych. Rozwiązanie to jest podobne do układu z konwerterem adresów [4] z tą różnicą, że kody klas łańcuchów pseudorównoważnych są pobierane z pamięci sterującej.

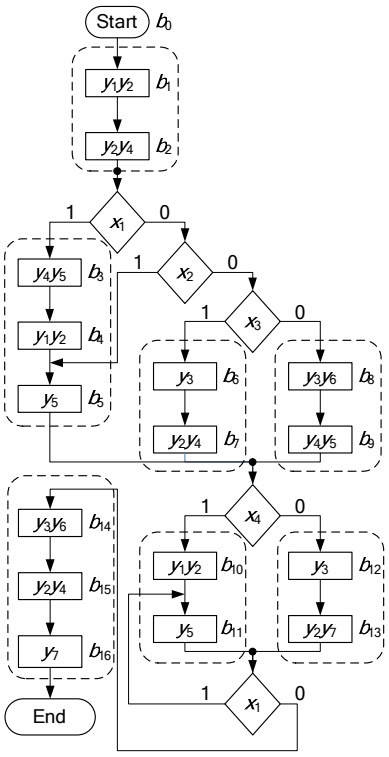
2. Sieć działań

Jedną z możliwych form reprezentacji algorytmu sterującego jest sieć działań (ang. Graph Scheme of Algorithm) GSA [1]. Przykładową sieć działań przedstawiono na rysunku 1. Sieć działań składa się ze zbioru wierzchołków $B=\{b_0, b_E\}\cup B_O\cup B_C$ oraz zbioru łuków $E=\{\langle b_i, b_j\rangle \mid b_i,b_j\in E\}$. Wierzchołek b_0 jest wierzchołkiem początkowym, wierzchołek b_E wierzchołkiem końcowym, zbiór B_O jest zbiorem wierzchołków operacyjnych, natomiast zbiór B_C jest zbiorem wierzchołków warunkowych. Wierzchołek operacyjny $b_i\in B_O$ zawiera zestaw mikrooperacji $Y(b_i)\subseteq Y$, gdzie $Y=\{y_1,\ldots,y_N\}$ jest zbiorem mikrooperacji realizowanych przez system. Wierzchołek warunkowy $b_j\in B_C$ reprezentuje element ze zbioru warunków logicznych odpowiadających wejściom układu $X=\{x_1,\ldots,x_M\}$.

Sieć działań Γ nazywa się *liniową* (Linear GSA – LGSA), jeśli liczba wierzchołków operacyjnych $|B_O|$ przekracza 75% wszystkich wierzchołów sieci [2].

Łańcuch bloków operacyjnych (Operational Linear Chain – OLC) jest sekwencją wierzchołków operacyjnych $\langle b_{g1},\ldots,b_{gFg}\rangle$,

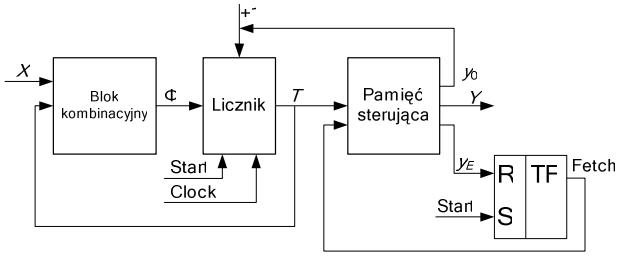
dla których istnieje łuk $\langle b_{gi}, b_{gi+1}\rangle$ dla każdej pary sąsiednich elementów (i=1,…,F_{g-1}). Każdy łańcuch α_g posiada jedno wyjście oraz skończoną liczbę wejść. Formalną definicję łańcucha OLC można znaleźć na przykład w [2, 4]. Na rysunku 1 łańcuchy bloków pseudorównoważnych zaznaczono linią przerywaną.



Rys. 1. Sieć działań Γ
Fig. 1. Graph scheme of algorithm Γ

3. Mikroprogramowany układy sterujący

Podstawową strukturę mikroprogramowanego układu sterującego przedstawiono na rysunku 2.



Rys. 2. Struktura CMCU U_1
Fig. 2. Strucuture of CMCU U_1

Układ CMCU U_1 składa się z bloku kombinacyjnego odpowiedzialnego za przygotowanie adresu skoku pomiędzy poszczególnymi łańcuchami OLC, bloku licznika przechowującego adres bieżącej mikroinstrukcji oraz bloku pamięci sterującej, w której zawarte są mikroinstrukcje wysyłane do ścieżki danych.

Uruchomienie układu następuje poprzez podanie wysokiego stanu na wejście *Start,* wtedy wraz z nadejściem narastającego zbocza sygnału zegarowego następuje zmiana zawartości licznika. Jeśli sygnał y_0=1 to wartość licznika jest zwiększana o 1, co odpowiada bezwarunkowemu przejściu do następnego wierzchołka sieci działań w ramach łańcucha OLC. W przypadku gdy y_0=0 to zawartość licznika jest ustalana przez układ kombinacyjny na podstawie stanu wejść X oraz bieżącego adresu mikroinstrukcji T. Zmiana zawartości licznika powoduje wystawienie przez pamięć sterującą nowej wartości na wyjściach Y oraz y_0.

4. Mikroprogramowany układ sterujący z rozszerzonym formatem mikroinstrukcji

W podstawowej strukturze mikroprogramowanego układu sterującego funkcja przejść zależy od stanu wejść oraz bieżącego adresu mikroinstrukcji:

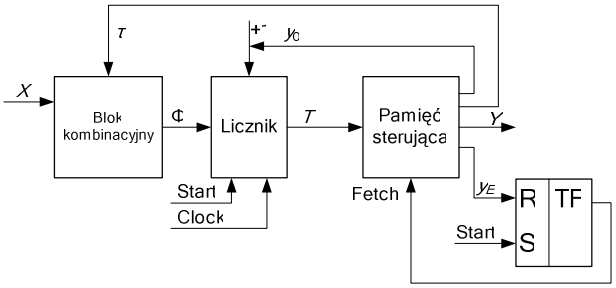
$$\Phi=\Phi(T, X). \tag{1}$$

Funkcję (1) można uprościć poprzez zgrupowanie łańcuchów OLC w klasy łańcuchów pseudorównoważnych. Dwa łańcuchy OLC α_i, $\alpha_j \in C$ są pseudorównoważne (pseudoequivalent OLC – POLC) jeżeli ich wyjścia są połączone z wejściem tego samego wierzchołka [5]. Dzięki wprowadzeniu podziału łańcuchów OLC na klasy POLC $\Pi=\{P_1,\ldots,P_I\}$ funkcja (1) może być wyrażona następująco:

$$\Phi=\Phi(K, X), \tag{2}$$

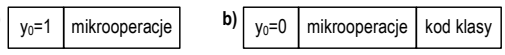
gdzie K jest kodem klasy $P_i \in \Pi$.

Kody klas POLC można uzyskać przez konwersję adresu mikroinstrukcji T za pomocą konwertera adresów [4]. Jednak zastosowanie bloku konwertera zwiększa rozmiar części kombinacyjnej układu. Innym rozwiązaniem pozwalającym na utrzymanie możliwie najmniejszego rozmiaru części kombinacyjnej jest umieszczenie kodów klas POLC w pamięci sterującej. Układ CMCU U_1 po modyfikacji przedstawiono na rysunku 3.



Rys. 3. Struktura CMCU U_2
Fig. 3. Strucuture of CMCU U_2

Umieszczenie kodów klas POLC w pamięci sterującej pociąga za sobą zmianę formatu mikroinstrukcji. Podstawowy format (Rys. 4a) zostaje rozszerzony o dodatkowe pole z kodem klasy $K(P_i)$ (Rys. 4b).



Rys. 4. Początkowy (a) i zmodyfikowany (b) format mikroinstrukcji
Fig. 4. Initial (a) and modified (b) microinstruction format

Modyfikacja formatu powoduje wydłużenie słowa pamięci, jednak nie zawsze wiąże się to ze wzrostem liczby zajmowanych przez układ bloków EMB dzięki wykorzystaniu zmiennej organizacji bloków EMB w układach FPGA. Możliwości konfiguracji osadzonych pamięci w układach serii Virtex II przedstawiono w tabeli 1.

Tab. 1. Możliwe warianty konfiguracji osadzonych pamięci w układach serii Xilinx Virtex II [5]
Tab. 1. Embedded block configurations modes in Xilinx Virtex II devices [5]

16K x 1 bit	2K x 9 bitów
8K x 2 bity	1K x 18 bitów
4K x 4 bity	512 x 36 bitów

Na przykład jeśli mikrooperacje zajmują 11 bitów, to podczas syntezy zostanie wybrany blok pamięci w wariancie 1K*18 bitów. Oznacza to, że pozostałe 7 bitów może być przeznaczonych na przechowywanie dodatkowych informacji takich jak np. kody klas POLC oraz dodatkowe zmienne sterujące (y_0, y_E).

Metoda syntezy układu U_2 w porównaniu do metody syntezy U_1 wymaga dodatkowo wyznaczenia klas łańcuchów POLC oraz ich zakodowania. Poszczególne kroki metody przedstawiają się następująco:

1. Wyznaczenie łańcuchów OLC, oraz klas łańcuchów POLC na podstawie sieci działań Γ.
2. Zaadresowanie mikroinstrukcji w ramach łańcuchów OLC z użyciem adresowania naturalnego.
3. Przypisanie kodów do klas łańcuchów POLC.
4. Przygotowanie zawartości pamięci sterującej.
5. Przygotowanie tabeli przejść.
6. Implementacja układu kombinacyjnego z użyciem tablic LUT oraz pamięci z wykorzystaniem bloków EMB.

5. Przykład zastosowania metody

Dla lepszego zobrazowania poszczególnych etapów syntezy CMCU U_2 wykorzystana zostanie sieć działań Γ (Rys. 1).

Dla sieci Γ zbiór łańcuchów OLC jest określony jako $C=\{\alpha_1,\ldots,\alpha_7\}$, gdzie $\alpha_1=\langle b_1, b_2\rangle$, $\alpha_2=\langle b_3, b_4, b_5\rangle$, $\alpha_3=\langle b_6, b_7\rangle$, $\alpha_4=\langle b_8, b_9\rangle$, $\alpha_5=\langle b_{10}, b_{11}\rangle$, $\alpha_6=\langle b_{12}, b_{13}\rangle$, $\alpha_7=\langle b_{14}, b_{15}, b_{16}\rangle$. Zbiór klas POLC $\Pi=\{P_1, P_2, P_3\}$, gdzie $P_1=\{\alpha_1\}$, $P_2=\{\alpha_2, \alpha_3, \alpha_4\}$, $P_2=\{\alpha_5, \alpha_6\}$ oznacza, to że kod klas ma długość 2 bitów.

Sieć działań Γ składa się z 16 mikroinstrukcji o długości 7 bitów. Słowo w pamięci sterującej będzie zatem miało długość 11 bitów: 7 bitów zajmują mikrooperacje, 2 bity kody klas łańcuchów POLC, 2 bity zmienne y_0, y_E. Zawartość pamięci po wykonaniu adresowania mikroinstrukcji przedstawiono w tabeli 2.

Tab. 2. Zawartość pamięci sterującej (dla wierzchołków $b_1,\ldots,b_5$ i b_{16})
Tab. 2. Content of control memory (vertexes $b_1,\ldots,b_5$ and b_{16})

Adres	y_0	y_E	$K(P_i)$	mikro-operacje	wierz-chołek
0000	1	0	-	1100000	b_1
0001	0	0	00	0101000	b_2
0010	1	0	-	0001100	b_3
0011	1	0	-	1100000	b_4
0100	0	0	01	0000100	b_5
1111	1	1	-	0000001	b_{16}

Tabelę przejść (Tab. 3) można sformułować na podstawie przejść między klasami POLC, które dla sieci działań Γ są określone następująco:

$$P_1 \rightarrow b_3x_1 \vee b_5\neg x_1x_2 \vee b_6\neg x_1\neg x_2x_3 \vee b_8\neg x_1\neg x_2\neg x_3$$
$$P_2 \rightarrow b_{10}x_4 \vee b_{12}\neg x_4$$
$$P_3 \rightarrow b_{11}x_1 \vee b_{14}\neg x_1.$$

Tab. 3. Tabela przejść
Tab. 3. Transitions table

P_i	$K(P_i)$	b_q	$A(b_q)$	X_h	Φ_h	h
P_1	00	b_3	0010	x_1	D_2	1
		b_5	0100	$\neg x_1x_2$	D_3	2
		b_6	0101	$\neg x_1\neg x_2x_3$	D_2D_4	3
		b_8	1000	$\neg x_1\neg x_2\neg x_3$	D_4	4
P_2	01	b_{10}	1010	x_4	D_1D_3	5
		b_{12}	1100	$\neg x_4$	D_1D_2	6
P_3	10	b_{11}	1011	x_1	$D_1\,D_3D_4$	7
		b_{14}	1110	$\neg x_1$	$D_1\,D_2D_3$	8

6. Badania eksperymentalne

Badania skuteczności zaproponowanej metody wykonano na podstawie 19 przykładów liniowych sieci działań [4] zapisanych w formacie KISS [6]. Modele układów CMCU w języku VHDL wygenerowano w oparciu o autorskie oprogramowanie ABSYNTH, a następnie poddano syntezie w systemie CAD Xilinx ISE (Wer. 10.2). Syntezę wykonano dla układów FPGA serii Virtex II Pro (xc2vp30-5-ff896) z optymalizacją powierzchni. Rezultaty badań przedstawiono w tabeli 4.

W przypadku wszystkich przykładów uzyskano zmniejszenie rozmiaru części adresowej CMCU U_2 względem podstawowego modelu CMCU U_1. Średnio rozmiar układu dla badanych przykładów został zmniejszony blisko o połowę, przy wykorzystaniu tej samej liczby przerzutników oraz jednego bloku pamięci BRAM.

Tab. 4. Wyniki syntezy
Tab. 4. Synthesis results

Test	We	Wy	µI	CMCU U_1		CMCU U_2		Redukcja U_2/U_1	
				Slice / LUT	Czas cyklu [ns]	Slice / LUT	Czas cyklu [ns]	Slice / LUT [%]	Czas cyklu [%]
mk01	10	7	86	42/74	7,07	20/36	5,88	48/49	83
mk02	10	7	90	40/70	8,92	10/19	4,49	25/27	50
mk03	7	8	49	29/51	5,01	12/23	3,40	41/45	68
mk04	9	8	57	42/73	5,10	18/32	3,91	43/44	77
mk05	10	9	55	34/59	5,48	14/27	3,91	41/46	71
mk06	8	11	59	25/47	7,70	16/30	4,40	64/64	57
mk07	11	8	71	50/89	12,02	22/41	5,45	44/46	45
mk08	7	7	50	28/49	7,21	14/25	3,48	50/51	48
mk09	7	7	57	40/66	5,48	20/37	3,88	50/56	71
mk10	13	11	93	47/83	12,20	22/38	6,11	47/46	50
mk11	8	8	49	36/65	6,07	18/31	5,78	50/48	95
mk12	11	8	72	43/75	11,17	19/34	5,70	44/45	51
mk13	6	7	38	22/38	6,42	11/21	3,51	50/55	55
mk14	8	7	54	24/42	5,42	18/32	3,54	75/76	65
mk15	8	10	69	32/57	7,07	17/28	5,91	53/49	84
mk16	6	10	72	29/52	9,88	15/28	5,38	52/54	54
mk17	9	7	73	47/82	10,43	18/32	6,55	38/39	63
mk18	6	11	52	31/56	5,43	18/30	3,80	58/54	70
mk19	6	7	47	32/57	5,73	15/27	4,27	47/47	74

7. Wnioski

Umieszczenie kodów klas łańcuchów POLC w pamięci sterującej pozwala zredukować rozmiar części kombinacyjnej układu sterującego, dzięki lepszemu wykorzystaniu osadzonych bloków pamięci. Pomimo wydłużenia mikroinstrukcji w pamięci sterującej układu U_2 w wielu przypadkach nie spowoduje to zwiększenia liczby wykorzystywanych bloków EMB w porównaniu do układu U_1.

Badania eksperymentalne wykonane dla układów FPGA serii Virtex II Pro potwierdzają, że średnia redukcja rozmiaru układu może sięgać około 50% przy wykorzystaniu tej samej liczby przerzutników oraz osadzonych bloków pamięci. Jednocześnie uproszczenie części kombinacyjnej przekłada się na poprawę parametrów czasowych układu. Dla testowanych sieci działań uzyskano skrócenie czasu cyklu średnio o około 30%.

Dalsze prace autorów będą skupiały się nad sprawdzeniem zaproponowanej metody dla układów typu CPLD oraz nad sprawdzeniem skuteczności metody dla sieci działań nie mających charakteru liniowego.

8. Literatura

[1] Baranov S. I.: Logic synthesis of Control Automata. Boston, Kluwer Academic Publishers, 1994.
[2] Alexander Barkalov, Larysa Titarenko: Logic synthesis for compositional microprogram control units. Springer-Verlag, Berlin 2008.
[3] Solovjev V. V.: Design of digital systems using the programmable logic integrated circuits. Moscow, Hot Line-Telecom, 2001.
[4] M. Kołopieńczyk: Application of address converter for decreasing memory size of compositional microprogram control unit with code sharing. University of Zielona Góra Press, Zielona Góra, 2008.
[5] Virtex-II Platform FPGAs: Complete Data Sheet. v3.5, Nov. 5, 2007
[6] S. Yang: Logic Synthesis and Optimization Benchmarks User Guide Version 3.0. Tech. Rep., Microelectr. Center of North Carolina, 1991.

Artykuł recenzowany